Eventos en LitElement

Miguel Angel Alvarez
@midesweb

ESCUELA IT

En esta sesión:

- Maneras de asociar manejadores de eventos
- Disparar eventos personalizados
- Capturar eventos desde en JS "nativo"

Miguel Angel Alvarez
@midesweb

ESCUELA IT

Miguel Angel Alvarez

miguel@desarrolloweb.com
@midesweb

ESCUELA IT

ASOCIAR
manejadores de evento

Miguel Angel Alvarez
@midesweb

ESCUELA IT

Desde un template

Es la manera "declarativa" que ya conocemos.

```
@click="${this.doClick}"
```

En el template declaramos el manejador de evento

```
doClick() {
  this.checked = !this.checked;
}
```

El manejador de evento se implementa como método del componente

Miguel Angel Alvarez
@midesweb

ESCUELA IT

De manera imperativa

Desde el constructor o desde firstUpdated() con addEventListener().

Cuidado especial para poder acceder a la variable "this".

Sería la manera posible de asignar un evento a la etiqueta host

```
constructor() {
  super();
  this.checked = false;
  this.addEventListener('click', e => {
    console.log('has hecho clic y checked vale ', this.checked);
  });
}
```

Miguel Angel Alvarez
@midesweb

ESCUELA IT

En connectedCallback

En ocasiones querrás asociar un evento cuando el componente se inserta en la página y quitarlo cuando el componente se elimina.

```
connectedCallback() {
  super.connectedCallback();
  document.addEventListener('click', this.doDocumentClick);
}
disconnectedCallback() {
  document.removeEventListener('click', this.doDocumentClick);
  super.disconnectedCallback();
}
```

Miguel Angel Alvarez
@midesweb

En conr

En ocasiones q
en la página y

Aquí vamos a tener un problema con el valor de "this", porque dentro del manejador valdrá `document`

```
constructor() {
  super();
  this.opened = false;
  this.doDocumentClick = this.close.bind(this);
}
```

```
connectedCallback() {
  super.connectedCallback();
  document.addEventListener('click', this.doDocumentClick);
}
disconnectedCallback() {
  document.removeEventListener('click', this.doDocumentClick);
  super.disconnectedCallback();
}
```

Miguel Angel Alvarez
@midesweb

ESCUELA IT

Objeto evento

El objeto evento se recibe en cualquier manejador. Este objeto es nativo de Javascript y ofrece mucha información útil sobre el evento que se ha producido.

```
lookForEnter(e) {
  let keycode = (e.keyCode ? e.keyCode : e.which);
  if (keycode == '13') {
      console.log('Has pulsado enter!')
  }
}
```

Miguel Angel Alvarez
@midesweb

EVENTOS personalizados

Miguel Angel Alvarez
@midesweb

ESCUELA IT

Eventos personalizados

Son eventos creados por nosotros mismos, que se disparan en el momento que ocurren cosas con los componentes.

```
doClick() {
  this.checked = !this.checked;
  this.dispatchEvent(new CustomEvent('switch-changed', {
    detail: {
      checked: this.checked
    }
  }));
}
```

La gestión de eventos personalizados se realiza enteramente con Javascript nativo.

Con "detail" podemos enviar cualquier dato al componente que escuche el evento

Miguel Angel Alvarez
@midesweb

ESCUELA IT

Capturar eventos personalizados

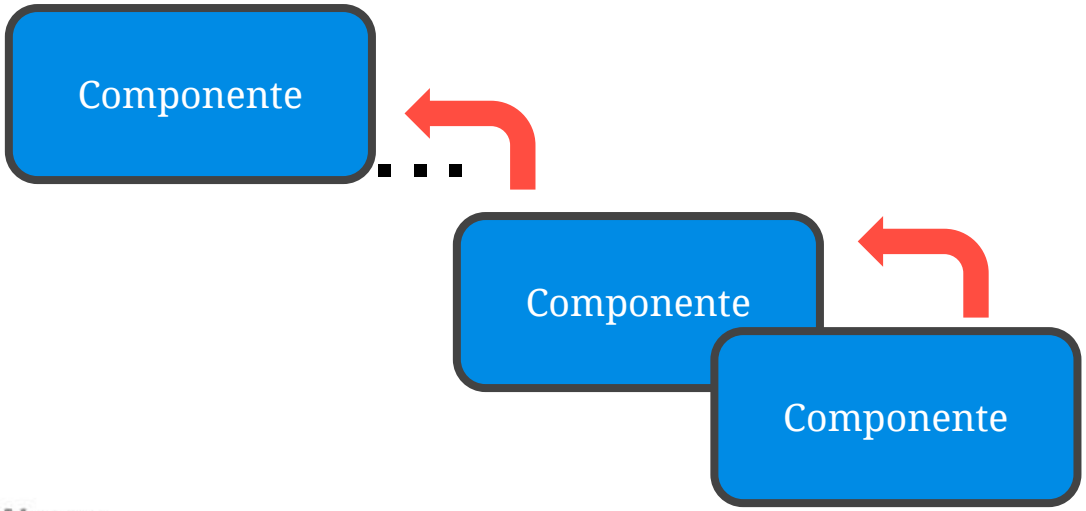
Se asocian manejadores igual que se hace con los eventos nativos.

```
<my-switch
  ?checked="${this.task.completed}"
  @switch-changed="${this.taskChange}"
></my-switch>
```

Miguel Angel Alvarez
@midesweb

Burbuja de eventos

La burbuja de eventos es un comportamiento nativo para los eventos del navegador.



Miguel Angel Alvarez
@midesweb

ESCUELA IT

Burbuja de eventos

Para nuestros eventos personalizados tenemos que activarla. Si no lo hacemos, el evento sólo llegará al padre.

Las propiedades bubbles y composed permiten que el evento llegue a padres, abuelos...

```
doClick() {
  this.checked = !this.checked;
  this.dispatchEvent(new CustomEvent('switch-changed', {
    bubbles: true,
    composed: true,
    detail: {
      checked: this.checked
    }
  }));
}
```

Miguel Angel Alvarez
@midesweb

ESCUELA IT

CAPTURAR EVENTOS desde js "nativo"

Miguel Angel Alvarez
@midesweb

ESCUELA IT

Usar addEventListener

Como los eventos se manejan de manera nativa, cuando usamos un componente en una página HTML, es posible añadir manejadores de eventos, incluso personalizados, como habitualmente en Javascript.

```
<my-switch id="mySwitch1"></my-switch>
```

```
<script>
document.getElementById('mySwitch1').addEventListener('switch-changed', (e) => {
  console.log('cambiado el valor del switch', e);
});
</script>
```

Miguel Angel Alvarez
@midesweb

ESCUELA IT

GRACIAS!!

Miguel Angel Alvarez
@midesweb

ESCUELA IT